

IN THE SPECIFICATION

Please replace the paragraph starting on page 2, line 21 with the following:

A1
The brute force approach such as that used by previous versions of the Shiva Virtual Private Networking (VPN) Manager, one component of a suite of graphical network management applications from Intel Corporation, the assignee of the present invention, is illustrated in **Figure 1**. First, the configuration file is downloaded from the remote device **110** of process 100 into the GUI application. Then, the user modifies the configuration file using the GUI application **120**. Finally, the process is reversed by uploading the modified configuration file back to the remote device **130**. The modified configuration file then refreshes the state of the CK with configuration changes that reflect the user modifications.

Please replace the paragraph starting on page 5, line 18 with the following:

A2
Referring now to **Figure 2**, wherein a block diagram illustrating the relationship of the GUI, CUI, and CK of the GDMVC method and apparatus in accordance with one embodiment is shown. As illustrated, the GUI **210** and CUI **220** form a self-contained application 200, with the CUI **220** running under the GUI **210**. Since the CUI **220** is not running on an actual remote device, as in the prior art device management systems, it is referred to as a virtual console, hence the name of the present invention. The use of a virtual console avoids the difficulties inherent in attempting to interact directly with the remote device's CUI, such as latency problems (i.e. delays in refreshing the contents of the CK), and resource contention (i.e. when more than one user

A2 attempts to update the configuration at the same time, thereby creating a potential lockout situation).

Please replace the paragraph starting on page 9, line 13 with the following:

A3 ¶ As shown, in one embodiment the latter two steps are accomplished by looping through the command list one at a time until the end of the list is reached. Specifically, the process involves seeking to the first command in the list 315, and if the end of the list has not yet been reached 320, then seeking again to the first registered command 330, and if the end of the list of registered commands has not yet been reached 335, then determining whether the command matches the registered command. If the command does not match the registered command, the process is repeated by seeking to the next registered command 350. Once a matching registered command is found 345 the associated graphical component is identified 355. If an instance of the associated graphical component does not yet exist, then one is first instantiated 360 and then initialized 365. Otherwise, the existing instance of the associated graphical component is simply initialized 365. This process is repeated by seeking to the next command in the list 340 until the list is exhausted 325. In another embodiment, an alternative method of comparing the list of commands to the command register is performed using a randomly accessible data repository for the comprehensive command register and associated graphical component and using a direct lookup by a command identifier, thereby eliminating the need for the looping logic.

Please replace the paragraph starting on page 11, line 4 with the following:

A4 ¶ With reference to Figure 4, the first step in the process 400 is to send 410 the configuration command associated with the graphical component to the CUI 220. In one embodiment this is accomplished by sending the command string representing the

A4
command to the CUI 220 using the C prototype SendCommand and the lpszCmd pointer. If the CUI 220 replies with an error message, the text of that message is returned 420 to the GUI 210 using the lpszMSG pointer of SendCommand. The GUI 210 displays the error message to the user 430 and any graphical component-specific recovery from the error is performed 440. For example, in the case of a configuration command associated with an edit control graphical component, the GUI 210 sets the input focus back to that edit control so that the user can correct the error. If the CUI 220 detects no error, the CUI 220 next determines whether the configuration command sent by the GUI 210 is interdependent with any other configuration commands 450. If so, the GUI 210 is refreshed using the standardized manner of refreshing the GUI 300 described in Figure 3 to reflect any changes in the CK 230 that may have resulted from the interdependencies.
